

FormatMessage

Vulnerable to string formatting issues

Sean Barnum, Cigital, Inc. [vita¹]

Copyright © 2007 Cigital, Inc.

2007-03-22

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5575 bytes

Attack Category	<ul style="list-style-type: none">• Malicious Input• Path spoofing or confusion problem								
Vulnerability Category	<ul style="list-style-type: none">• Buffer Overflow• Format string								
Software Context	<ul style="list-style-type: none">• String Management• National Language Support								
Location	<ul style="list-style-type: none">• winbase.h								
Description	<p>The FormatMessage function formats a message string. The function requires a message definition as input. The message definition can come from a buffer passed into the function. It can come from a message table resource in an already-loaded module. Or the caller can ask the function to search the system's message table resource(s) for the message definition. The function finds the message definition in a message table resource based on a message identifier and a language identifier. The function copies the formatted message text to an output buffer, processing any embedded insert sequences if requested.</p> <p>This function takes a lot of overloaded and confusing arguments that can potentially be misused.</p> <p>The primary weakness is a string formatting issue that can lead to a buffer overflow. Avoid using the "%s" in format string. On Windows 95/98/Me, no single insertion string may exceed 1023 characters in length.</p> <p>Note: This function takes a max. output string size, so it may not be vulnerable to format string attack.</p>								
APIs	<table border="1"><thead><tr><th>Function Name</th><th>Comments</th></tr></thead><tbody><tr><td>FormatMessage</td><td>Src: 7 variable; Fmt: 1</td></tr><tr><td>FormatMessageA</td><td>Src: 7 variable; Fmt: 1</td></tr><tr><td>FormatMessageW</td><td>Src: 7 variable; Fmt: 1</td></tr></tbody></table>	Function Name	Comments	FormatMessage	Src: 7 variable; Fmt: 1	FormatMessageA	Src: 7 variable; Fmt: 1	FormatMessageW	Src: 7 variable; Fmt: 1
Function Name	Comments								
FormatMessage	Src: 7 variable; Fmt: 1								
FormatMessageA	Src: 7 variable; Fmt: 1								
FormatMessageW	Src: 7 variable; Fmt: 1								

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

Method of Attack	An attacker could use the string formatting features to inject malicious input to overflow the buffer.		
Exception Criteria			
Solutions	Solution Applicability	Solution Description	Solution Efficacy
		If dwFlags contains FORMAT_MESSAGE_FROM_STRING, then lpSource contains a printf-style format string. For each "%n!s!" reference within the message text, you should include precision modifiers to be sure that the destination buffer is not exceeded.	
Signature Details	DWORD FormatMessage(DWORD dwFlags,LPCVOID lpSource,DWORD dwMessageId,DWORD LanguageId,LPTSTR lpBuffer,DWORD nSize,va_list* Arguments);		
Examples of Incorrect Code	<pre> /* If precision modifiers are not used, there is a chance that there could be buffer */ /* overflow */ #include <windows.h> void ErrorExit(LPTSTR lpSzFunction) { TCHAR szBuf[80]; LPVOID lpMsgBuf; DWORD dw = GetLastError(); FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER FORMAT_MESSAGE_FROM_SYSTEM, NULL, dw, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPTSTR) &lpMsgBuf, 0, NULL); wsprintf(szBuf, </pre>		

	<pre> "%s failed with error %d: %s", lpSzFunction, dw, lpMsgBuf); MessageBox(NULL, szBuf, "Error", MB_OK); LocalFree(lpMsgBuf); ExitProcess(dw); } </pre>				
Examples of Corrected Code					
Source References	<ul style="list-style-type: none"> • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/formatmessage.asp² • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/formatmessage.asp³ • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/retrieving_the_last_error_code.asp⁴ 				
Recommended Resource					
Discriminant Set	<table border="1"> <tr> <td>Operating System</td> <td> <ul style="list-style-type: none"> • Windows </td> </tr> <tr> <td>Languages</td> <td> <ul style="list-style-type: none"> • C • C++ </td> </tr> </table>	Operating System	<ul style="list-style-type: none"> • Windows 	Languages	<ul style="list-style-type: none"> • C • C++
Operating System	<ul style="list-style-type: none"> • Windows 				
Languages	<ul style="list-style-type: none"> • C • C++ 				

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com¹.

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. <mailto:copyright@cigital.com>